NPS55-89-003

# NAVAL POSTGRADUATE SCHOOL

Monterey, California

DEPARTMENT OF THE NAVY ★ NAVAL POSTGRADUATE SCHOOL ★

FINITE METHODS FOR A NONLINEAR ALLOCATION PROBLEM

ALAN R. WASHBURN

APRIL 1989

Approved for public release; distribution is unlimited.

ared for:

ef of Naval Operations
ington, D.C.

NAVAL POSTGRADUATE SCHOOL,
MONTEREY, CALIFORNIA

The work reported herein was supported in part with funds provided from the Chief of Naval Operations.

This report was prepared by:

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

| | |
|---|---|
| 1a REPORT SECURITY CLASSIFICATION: UNCLASSIFIED | 1b RESTRICTIVE MARKINGS |
| 2a SECURITY CLASSIFICATION AUTHORITY | 3 DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited. |
| 2b DECLASSIFICATION/DOWNGRADING SCHEDULE | |
| 4 PERFORMING ORGANIZATION REPORT NUMBER(S): NPS55-89-03 | 5 MONITORING ORGANIZATION REPORT NUMBER(S) |
| 6a NAME OF PERFORMING ORGANIZATION: Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable): Code 55 | 7a NAME OF MONITORING ORGANIZATION |
| 6c ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000 | 7b ADDRESS (City, State, and ZIP Code) |
| 8a NAME OF FUNDING/SPONSORING ORGANIZATION: Naval Postgraduate School; 8b OFFICE SYMBOL (If applicable) | 9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: O & MN, Direct Funding |
| 8c ADDRESS (City, State, and ZIP Code): Monterey, CA. 93943 | 10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO |

11 TITLE (Include Security Classification)
Finite Methods for a Nonlinear Allocation Problem

12 PERSONAL AUTHOR(S)
Alan R. Washburn

| 13a TYPE OF REPORT | 13b TIME COVERED | 14 DATE OF REPORT (Year, Month, Day) | 15 PAGE COUNT |
|---|---|---|---|
| technical | FROM Oct 87 TO Sep 88 | April 1989 | 20 |

16 SUPPLEMENTARY NOTATION

| 17 COSATI CODES | | | 18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number) |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | allocation, nonlinear |
| | | | |
| | | | |

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

We consider the problem of allocating m resources to n tasks. The potential effectiveness, $y_j$ for the $j^{th}$ task is a linear function of the allocations to that task, and the objective function is a convex function $f(y_1, \ldots, y_n)$ to be minimized. Two finite algorithms for obtaining an optimal solution are proposed and evaluated.

| | |
|---|---|
| 20 DISTRIBUTION/AVAILABILITY OF ABSTRACT: [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS | 21 ABSTRACT SECURITY CLASSIFICATION: Unclassified |
| 22a NAME OF RESPONSIBLE INDIVIDUAL: A. R. Washburn | 22b TELEPHONE (Include Area Code): (408) 646-3127; 22c OFFICE SYMBOL: 55Ws |

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

☆ U.S. Government Printing Office: 1986—606-243

# Finite Methods for a Nonlinear Allocation Problem

Alan Washburn

## 1. Introduction

We consider here finite algorithms for solving the nonlinear program P1:

$$\text{minimize } f(y_1,\ldots,y_n)$$

$$\text{subject to} \quad \sum_{i=1}^{m} E(i,j)x_{ij}=y_j; \quad 1\le j\le n,$$

$$\sum_{j=1}^{n} x_{ij}=b_i; \quad 1\le i\le m,$$

$$x_{ij}\ge 0;\ 1\le i\le m,\ 1\le j\le n.$$

It is assumed in P1 that $b_i\ge 0$ for all i and that $E(i,j)\ge 0$ for all (i,j), with at least one positive E(i,j) in each row i. It is further assumed that f(y) is continuous and decreasing in each of its arguments on some convex set S that includes the nonnegative orthant of $R^n$. y represents the vector $(y_1,\ldots,y_n)$, and similarly x will represent the collection of all mn of the $x_{ij}$, etc. P1 is a special case of the nonlinear network problem, a problem that is already known to possess sufficient special structure to make the development of special purpose software attractive(Ahlfeld, Dembo, Mulvey, and Zenios(1987)). Our intention is to specialize P1 even further by making restrictive assumptions about f(y). The components of y will be called "potentials", one for each column of the matrix (E(i,j)). Potentials are not logically necessary in defining P1, since the expressions defining y could simply be substituted into f(y), but the generous notation will prove useful in the sequel. The components of x will be called "allocations."

P1 could be interpreted as a Search Theory problem by assuming that search is to be conducted over a fixed time period by m distinct types of searcher, with $b_i$ units of type i effort available over the period. Assume that a single target is located with probability $p_j$ in one of n regions, that region j has area $A_j$, and that the reward for finding the target in region j is $V_j$. Assume further that the amount of area swept by a searcher of type i in region j is $s_{ij}$. If $x_{ij}$ is the allocation of search effort of type i to region j, then the total area swept in region j is $\sum_{i=1}^{m} x_{ij}s_{ij}$. If the search in each area is "random" (Koopman(1956)), then the number of times a target in region j is detected is a Poisson random variable with mean $\left(\sum_{i=1}^{m} x_{ij}s_{ij}\right)/A_j=y_j$ (so let $E(i,j)=s_{ij}/A_j$), and the probability of no detections is $\exp(-y_j)$. The average "regret" in terms of value not found is then $f(y)=\sum_{j=1}^{n} V_jp_j\exp(-y_j)$. P1 is then the problem of allocating m resources to n regions in order to minimize the average undetected value. P1 can be thought of as a generalization from the case m=1, which already has specially tailored solution procedures (Washburn(1981)).

The US Air Force's Heavy Attack model (Clasen, Graves, and Lu(1974)) is also closely related to P1, with resources being aircraft sorties, areas being classes of targets, and potentials being "average number of targets killed were it not for the fact that some of the potential will be 'wasted' in attacking targets already dead". In general, P1 is applicable when n projects must be simultaneously undertaken by employing m resources. y is a vector measure of progress, and f(y) is some global, scalar measure of work not completed.

## 2. Basic Feasible Solutions

P1 has mn potentially positive allocations, but optimal solutions invariably have most of these being 0. The reason for this is that optimal solutions need never include cycles of positive allocations.

**Definition:** If a feasible allocation for P1 has an alternating sequence $j_1,i_1,j_2,i_2,\ldots,i_L,j_{L+1}$ of columns and rows all distinct except that $j_{L+1}=j_1$, and if $x_{ij}>0$ for every consecutive pair in the sequence ($(i,j)=(i_k,j_k)$ or $(i,j)=(i_k, j_{k+1})$ for some $k\leq L$), then the sequence is a **cycle**.

**Definition:** A feasible allocation for P1 is **conservative** if $x_{ij}=0$ whenever $E(i,j)=0$.

**Theorem 1:** Given the assumptions of section 1, P1 has an optimal, conservative solution with no cycles.
**Proof:** P1 has an optimal solution because the objective function regarded as a function of x is continuous on a compact set (Bazaraa and Shetty(1979)). An optimal solution x can easily be converted to a conservative optimal solution by shifting any offending allocation $x_{ij}$ from column j to some other column k where $E(i,k)>0$ (by assumption at least one such column k always exists for every i). Assume therefore that x is optimal and conservative, but that a cycle exists. In that case construct a modified solution x' by adding $\delta(i,j)$ to $x_{ij}$ for each consecutive pair (i,j) in the sequence, where $\delta(i,j)$ is defined as follows:

$$\delta(i_1,j_1)=z \text{ (an arbitrary real number)}$$
$$\delta(i_k,j_{k+1})=-\,\delta(i_k,j_k);\ 1\leq k\leq L$$
$$\delta(i_{k+1},j_{k+1})=-\,\delta(i_k,j_{k+1})E(i_k,j_{k+1})/E(i_{k+1},j_{k+1});\ 1\leq k\leq L-1.$$

The numbers $\delta(i,j)$ are constructed so that $\sum_{j=1}^{n} x'_{ij}=b_i$ for all i as before. Also

$\sum_{i=1}^{m} x'_{ij}E(i,j)=y_j$ as before, except that the latter sum for $j=j_1$ is increased by

$\delta(i_1,j_1)E(i_1,j_1)-\delta(i_L,j_1)E(i_L,j_1)$. This difference is proportional to z. If the proportionality constant is nonnegative, increase z until $x'_{ij}=0$ for some (i,j) in the cycle; there will be some such (i,j) because the numbers $\delta_{ij}$ are proportional to z with nonzero proportionality constants that alternate in sign. Otherwise, decrease z to achieve the same end. Since f() is decreasing in each variable, the x' solution is at least as good as the x solution, with one less positive variable. By repeating this operation if necessary, an optimal solution with no cycles can be found. QED

Since an optimal conservative solution with no cycles exists, it makes sense to search for the optimal solution amongst solutions with that property. A major advantage of such solutions is that there can be at most n+m–1 positive variables $x_{ij}$. To see this, consider the nonoriented bipartite graph G formed by connecting m "row nodes" to n "column nodes", with i being connected to j if and only if $x_{ij}>0$. Such a graph must consist of a number of connected components $T_1,\ldots T_K$, each of which is not connected to any of the others. Some of these components may contain a single node and no edges, but every node is in exactly one component. Since each component is connected and has no cycles, it is a tree (it is convenient to refer to even singleton components as "trees"), so G is a "forest". If $T_k$ has $r_k$ nodes, it must have exactly $r_k-1$ edges (Berge(1962)), even if $r_k=1$. Therefore the total number of edges is $n+m-K$, which is at most $n+m-1$.

The graph G described above will be called the "basis" of any conservative solution with no cycles. More generally,

**Definition:** Any nonoriented bipartite graph G with the following properties is a **basis:**

1) The nodes of G are all of the m+n row and column nodes.
2) All row nodes I for which $b_I>0$ have at least one incident edge.
3) If $E(i,j)=0$, then there is no edge (i,j).
4) There are no cycles.

The requirement that a solution should have no cycles is sufficient to establish that the corresponding basis is a forest, but it is not sufficient to establish a one-to-one relationship between bases and solutions. In order to establish such a relationship, further assumptions about f() are needed.

**Definition:** f() is **good** if it has the following properties:

1) f() is differentiable and decreasing on the interior of the convex set S described earlier. Let f'() be the gradient of f().

2) There exists a unique function $g(\mu)$ taking values in S such that $f'(g(\mu))+\mu=0$ for all $\mu>0$, where by $\mu>0$ we mean $\mu_j>0$ for $j=1,\ldots,n$.

3) If C is a nonempty subset of $\{1,\ldots,n\}$, if $\mu>0$, and if scalar $\rho>0$, then the equation $\sum_{j\varepsilon C}\mu_j g_j(z\mu)=\rho$ has a unique positive solution z. Furthermore z is a continuous function of $\rho$ for $\rho>0$.

In practice the methods introduced below will be attractive only if the solution z required in the last part is easily computed, since the equation will have to be solved many times. Here are two examples of good functions:

Example 1: $f(y)=\sum_{j=1}^{n} V_j\exp(-y_j)$, with $S=R^n$. This is the Search Theory function introduced earlier, except that $p_j$ has been incorporated into $V_j$. In this case $g_j(\mu)=\ln\left(V_j/\mu_j\right)$ for $j=1,\ldots,n$, and $\ln(z)=\left[\sum_{j\varepsilon C}\mu_j g_j(\mu)-\rho\right]\Big/\left[\sum_{j\varepsilon C}\mu_j\right]$.

Example 2: $f(y)=\sum_{j=1}^{n} 1/(1+y_j)$, with $S=\{y\colon y_j>-1 \text{ for } j=1,\ldots,n\}$. Here $g_j(\mu)=\left(\sqrt{v_j/\mu_j}-1\right)$ and $\sqrt{z}=\left[\sum_{j\varepsilon C}\sqrt{V_j\mu_j}\right]\Big/\left[\rho+\sum_{j\varepsilon C}\mu_j\right]$, well defined for $\rho>0$.

The Kuhn-Tucker conditions for P1 are that there should exist $\lambda,\mu,x$, and y such that

KT1) $\lambda_i \geq \mu_j E(i,j)$ for all i,j

KT2) $x_{ij} \geq 0$ for all i,j

KT3) $\lambda_i = \mu_j E(i,j)$ when $x_{ij} \neq 0$

KT4) $\sum_{i=1}^{m} x_{ij} E(i,j) = y_j$ for all j

KT5) $\sum_{j=1}^{n} x_{ij} = b_i$ for all i, and

KT6) $y = g(\mu)$.

The object now is to use the equality parts of the KT conditions (KT3–KT6) to set up a one-to-one correspondence between bases and solutions for good objective functions, after which solutions corresponding to bases will be called "basic." The vectors $\lambda$ and $\mu$ will each be called "multipliers." Basic solutions will have $x_{ij}=0$ for nonbasic edges (i,j), but the following theorem permits one exception to support the pivoting ideas of section 4.

**Theorem 2:** For any tree T let R(T) and C(T) be the row nodes and column nodes of T, respectively. Let G be a basis composed of trees $(T_1,\ldots,T_K)$, and suppose that $\lambda_i = \mu_j E(i,j)$ for all edges (i,j) of G. Then

a) If (x,y) solve KT4–KT5, and if all nonbasic allocations except for $x_{IJ}$ are zero, then for any component T of G,

$$\left[\sum_{j\varepsilon C(T)} \mu_j y_j\right] - \mu_J x_{IJ} E(I,J)\delta(C(T),J) = \left[\sum_{i\varepsilon R(T)} \lambda_i b_i\right] - \lambda_I x_{IJ}\delta(R(T),I), \qquad (1)$$

where now $\delta(W,w)$ is 1 if w is in the set W, otherwise 0. Furthermore,

b) If $(\lambda,\mu,y)$ is such that (1) holds for every component T of G, then KT4–KT5 have a unique solution x for which all nonbasic allocations except $x_{IJ}$ are zero.

**Proof of part a)** Multiply both sides of KT4 by $\mu_j$ and sum to obtain

$$\sum_{j\varepsilon C(T)} \sum_{i=1}^{m} \mu_j x_{ij} E(i,j) = \sum_{j\varepsilon C(T)} \mu_j y_j. \qquad (2)$$

Since $x_{ij}=0$ when $j\varepsilon C(T)$ unless either i=I or $i\varepsilon R(T)$,

$$\sum_{j\varepsilon C(T)} \sum_{i\varepsilon R(T)}^{\cdot} \mu_j x_{ij} E(i,j) + \mu_J x_{IJ} E(I,J)\delta(c(T),J) = \sum_{j\varepsilon C(T)} \mu_j y_j. \qquad (3)$$

But $\mu_j E(i,j) = \lambda_i$ when $j\varepsilon C(T)$ and $i\varepsilon R(T)$ by assumption, so

$$\sum_{i\varepsilon R(T)} \lambda_i \sum_{j\varepsilon C(T)} x_{ij} + \mu_J x_{IJ} E(I,J)\delta(c(T),J) = \sum_{j\varepsilon C(T)} \mu_j y_j. \qquad (4)$$

Since KT5 holds,

$$\sum_{j\varepsilon C(T)} x_{ij} = b_i - x_{IJ}\delta(\{i\},I) \text{ for all } i\varepsilon R(T). \qquad (5)$$

Part a) follows upon substituting (5) into (4).

**Proof of part b):** To avoid unnecessary complication assume $x_{IJ}=0$; otherwise $b_I$ and $y_J$ can be adjusted by subtracting $x_{IJ}$ and $x_{IJ}E(I,J)$, respectively. Consider any component T of G. We must show that there is exactly one way to assign allocations $x_{ij}$ to the edges of T that is consistent with KT4–KT5. This is trivial if T is a singleton; otherwise, since T is a tree, there must be at least 1 pendant (connected to exactly one other node) node. If this pendant node is row P, let row P be connected to column Q. Then $x_{PQ}$ is the only nonzero allocation in row P, and is therefore necessarily $b_P$ by KT5. Define y' by $y'_Q=y_Q-b_PE(P,Q)$, $y'_j=y_j$ for $j\neq Q$, and let T' be the tree remaining after node P and edge (P,Q) are deleted. Since $\lambda_P=\mu_QE(P,Q)$, $\mu_Qy'_Q=\mu_Qy_Q-\lambda_Pb_P$. Therefore $\sum_{j\varepsilon C(T')}\mu_jy'_j = \sum_{i\varepsilon R(T')}\lambda_ib_i$. If the pendant node is column Q, let it be connected to row P. Since $x_{PQ}$ is the only positive allocation in column Q, necessarily $x_{PQ}$ is $y_Q/E(P,Q)$ according to KT4. $x_{PQ}$ is well defined because $E(P,Q)>0$. Let $b'_P=b_P-x_{PQ}$, otherwise let $b'_i=b_i$ for $i\neq P$, and let T' be the tree resulting from deleting column Q and edge (P,Q). Since $\lambda_P=\mu_QE(P,Q)$, $\lambda_pb'_P=\lambda_Pb_P-\mu_Qy_Q$. Therefore $\sum_{j\varepsilon C(T')}\mu_jy_j = \sum_{i\varepsilon R(T')}\lambda_ib'_i$. Since in either case T' has one less node than T, and since the proposition is true for graphs with one node, the theorem is proved by induction. QED

Multipliers for which $\lambda_i=\mu_jE(i,j)$ on basic edges are easy to determine if there is no requirement for equation (1) to hold. The reason is that if $\lambda_i$ or $\mu_j$ is determined at any node, then the same thing is true of all neighbors of the node. Thus $\lambda$ and $\mu$ can be determined in any component T of G by assigning an arbitrary positive value to any node. Let $(\lambda^*,\mu^*)$ be such multipliers. Then any other set $(\lambda,\mu)$ with the same property is some scalar multiple of $(\lambda^*,\mu^*)$. Let $z>0$ be the multiple. Then z must be determined so that (1) holds. After substituting $g_j(z\mu^*)$ for $y_j$, (1) is:

$$\sum_{j\varepsilon C(T)}\mu^*_jg_j(z\mu^*) - \mu^*_Jx_{IJ}E(I,J)\delta(C(T),J) = \sum_{i\varepsilon R(T)}\lambda^*_ib_i - \lambda^*_Ix_{IJ}\delta(R(T),I). \qquad (6)$$

As long as $x_{IJ}\leq b_I$, the right hand side of (6) is nonnegative. Equation (6) will then have a unique positive solution z as long as f() is good. Given z, KT3–KT6 have the unique solution $(\lambda,\mu)=z(\lambda^*,\mu^*)$, $y=g(\mu)$, and x the unique solution of KT4–KT5. The x part is the desired basic solution corresponding to the basis G–hereafter the G-basic solution. Thus the unique G-basic solution can be easily determined as long as $x_{IJ}\leq b_I$.

A G-basic solution need not satisfy KT2, but if it does then it will be called a basic feasible solution and G will be called "feasible". Some basic feasible solution is optimal, so P1 could be solved in principle by examining all bases. In analogy with the Simplex Method, however, the object is to avoid this by utilizing a procedure that is directional in the sense of considering a sequence of feasible bases, each of which is better than the last. Two of these will be considered below. In each case we will prove only that the objective function sequence is nonincreasing, rather than decreasing, with equality being possible in degenerate cases. Rather than deal with the distracting issue of degeneracy here, we simply assume nondegeneracy for the next two sections. There is every reason to expect that the same techniques that handle degeneracy in the Simplex Method will also work here.

## 3. The Manifold Suboptimization Method

In this section nonbasic variables will always be 0.

A G-basic solution is the optimal solution of P2(G):

$$\begin{aligned} &\text{mimimize } f(y) \\ \text{subject to}\quad &\sum_{i=1}^{m} E(i,j)x_{ij}=y_j; \quad 1\leq j\leq n, \\ &\sum_{j=1}^{n} x_{ij}\leq b_i; \quad 1\leq i\leq m, \\ &x_{ij}=0 \text{ unless } (i,j)\varepsilon G. \end{aligned}$$

To prove this note that the G-basic solution solves KT3–KT6. But KT3–KT6 are the Kuhn Tucker conditions for P2(G), and the Kuhn Tucker conditions are sufficient for a global minimum because f() is convex as a function of x. P2(G) does not require allocations to be nonnegative, so the feasible space is not compact. Absent the constraints forcing nonbasic variables to be 0, P2(G) might lack a (finite) optimal solution. Nonetheless, the G-basic solution is optimal.

Given a basis G and a corresponding solution x feasible in P1, suppose KT1 is not satisfied for some (I,J). The natural computational step is to simply put (I,J) into G. Any cycle that forms (a cycle will form if and only if I and J are in the same tree) can always be profitably destroyed by removing some other edge $(K_1,L_1)$ determined as in Theorem 1. (I,J) would be $(i_1,j_1)$ in that theorem, and z would be positive because the reduced cost for (I,J) is negative. Let $G^1$ be G with (I,J) added and, if necessary, $(K_1,L_1)$ deleted. Let $x^{1+}$ be x if $(K_1,L_1)$ is not deleted, or otherwise the solution x' of Theorem 1. $x^{1+}$ is better than x and feasible, but is not in general the $G^1$-basic solution. The latter solution (call it $x^1$) is better than $x^{1+}$ because it is optimal in $P2(G^1)$ while $x^{1+}$ is merely feasible. $x^1$ is therefore basic and better than x. However, $x^1$ may not be feasible in P1.

If $x^1$ has one or more negative components, let $x^{2+}=x(1-\alpha)+x^1\alpha$, where $\alpha\varepsilon[0,1]$ is selected so that $x^{2+}$ is the closest nonnegative point to $x^1$. Let $(K_2,L_2)$ be an edge in $G^1$ whose allocation is driven to 0 at $x^{2+}$. Since f() is convex and $x^1$ is better than x, $x^{2+}$ is also better than x. $x^{2+}$ is also feasible in $P2(G^2)$, where $G^2$ is the basis obtained by deleting edge $(K_2,L_2)$ from $G^1$. Let $x^2$ be optimal in $G^2$, so that $x^2$ is better than $x^{2+}$. If $x^2$ also has negative components, continue to define $x^{3+}$,$G^3$, $x^3$, $x^{4+}$, etc., until finally $x^k$ is nonnegative, as must eventually happen because an edge is deleted from the basis at each step. $x^k$ is then basic, feasible (in P1), and (barring degeneracy) better than x. Since there are only finitely many bases, none of which can repeat because the objective function decreases at each step, the optimum solution must eventually be found in finitely many steps.

The above algorithm is sufficiently similar to Zangwill's (1970) method of Manifold suboptimization that the name has been retained, but there is a difference. The difference is that in Zangwill's method a G-basic, feasible solution that is not optimal would lead to next considering a modified basis G'=G+(I,J), where (I,J) is some edge at which the reduced cost is negative. Unfortunately G' may have a cycle, in which case P2(G') may not have a finite optimal solution. The method described above works because every manifold considered has a basis that is a forest of trees.

Manifold suboptimization essentially operates by increasing $x_{IJ}$ so much that the next basic solution may not be feasible, and then fixing the problem. If only feasible solutions are to be considered, then a test for the amount that $x_{IJ}$ can be increased without causing an infeasibility must be found. This is the subject of the next section.

## 4. The Extended Simplex Method

In this section it will be necessary to consider situations where nonbasic variables are temporarily positive, as in the Simplex Method. Assume that G is a feasible basis, but that the corresponding basic feasible solution is not optimal because KT1 is not satisfied for some (I,J). Then the reduced cost $\lambda_I - \mu_J E(I,J)$ is negative, so $x_{IJ}$ should be increased as in Manifold suboptimization. Since the generality will be needed later, suppose that the nonbasic variable $x_{IJ}$ is fixed at some level b between 0 and $b_I$, and that the balance equation (6) is satisfied for each component T of G by z=1 when $x_{IJ}=b$. If the nonbasic variable $x_{IJ}$ is increased from b to $b+\Delta$, let the solution of (6) for z be $Z(T,\Delta)$. If f() is a good function, $Z(T,\Delta)$ is a continuous, positive function of $\Delta$ with $Z(T,0)=1$. The corresponding multipliers $(\lambda(\Delta),\mu(\Delta))$ are the same as $(\lambda^*,\mu^*)$ except that multipliers for nodes in T are multiplied by $Z(T,\Delta)$. Let $y(\Delta)=g(\mu(\Delta))$, let $x(\Delta)$ be the unique solution of KT4–KT5 where only basic variables are nonzero except that nonbasic $x_{IJ}=b+\Delta$, let $\Delta'$ be the largest $\Delta$ such that $x(\Delta)\geq 0$, and let $x_{KL}(\Delta')=0$. In other words, basic variable (K,L) is the one first driven to 0 by increasing $\Delta$. If $x(\Delta)>0$ for all $\Delta\geq 0$, let $\Delta'$ and (K,L) be undefined. If (K,L) is defined and if its deletion from the basis breaks any cycle that introducing (I,J) might form, then the next basis simply replaces (K,L) with (I,J). (K,L) is not necessarily defined or part of a cycle even if it is defined, but Algorithm A below accounts for these possibilities. The input to the algorithm is a feasible basis $G^0$, associated multipliers $(\lambda^0,\mu^0)$, and a nonbasic edge (I,J) with $x_{IJ}=0$ and negative reduced cost. The output is a different feasible basis G' at least as good as $G^0$, together with associated multipliers $(\lambda',\mu')$. The basic idea of the algorithm is to increase $x_{IJ}$ in steps until finally the reduced cost is zero. If I and J are in the same tree, $x_{IJ}$ is increased in step A4 until some edge is deleted. Eventually this operation will put I and J into different trees. When I and J are in different trees, either an edge is deleted from one of the trees ($\Delta'<\Delta$ in step A5) or else (I,J) is finally added to the basis in the terminal step A9. Here is Algorithm A.

A1) Let k=0, $X^0=0$

A2) Let $\lambda^*=\lambda^k$, $\mu^*=\mu^k$, $G=G^k$, and $b=X^k$.

A3) Let $T_I$ be the tree in $G^k$ that includes I and $T_J$ be the tree that includes J. If $T_I\neq T_J$ go to A5.

A4) Let $T=T_I$. Determine $\Delta'$ by solving (6) for $Z(T,\Delta)$ and proceeding as in the paragraph above. Let (K,L) be the basic edge in T whose variable is first driven to 0, let $G^{k+1}=G^k-(K,L)$, $(\lambda^{k+1},\mu^{k+1})=(\lambda(\Delta'),\mu(\Delta'))$, and $X^{k+1}=X^k+\Delta'$. Go to A2.

A5) Let $T=T_I$. Determine $\Delta'$ by solving (6) for $Z(T,\Delta)$ and proceeding as in the paragraph above. Let (K,L) be the basic edge in T whose variable is first driven to 0. Let $(K_I,L_I)=(K,L)$ and $\Delta_I=\Delta'$.

A6) Let $T=T_J$. Determine $\Delta'$ by solving (6) for $Z(T,\Delta)$ and proceeding as in the paragraph above. If (K,L) is undefined let $\Delta_J=\infty$; otherwise, let $(K_J,L_J)=(K,L)$ and

$\Delta_J=\Delta'$.

A7) Let $\Delta^*$ be the smallest nonnegative solution of the equation $\lambda^*_I Z(T_I,\Delta)=\mu^*_J Z(T_J,\Delta)E(I,J)$, with $\Delta^*=\infty$ if there is no such solution.

A8) Let $\Delta'=\min\{\Delta_I,\Delta_J,\Delta^*\}$. Let $\lambda_i^{k+1}=\lambda_i^k Z(T_I,\Delta')$ for $i\varepsilon T_I$, or $\lambda_i^{k+1}=\lambda_i^k Z(T_J,\Delta')$ for $i\varepsilon T_J$, or otherwise $\lambda_i^{k+1}=\lambda_i^k$. Define $\mu^{k+1}$ similarly. Let $X^{k+1}=X^k+\Delta'$.

A9) If $\Delta'=\Delta^*$, set $G'=G^k+(I,J)$ and $(\lambda',\mu')=(\lambda^{k+1},\mu^{k+1})$. Stop.

A10) If $\Delta'=\Delta_I$, let $G^{k+1}=G^k-(K_I,L_I)$ and $k=k+1$. Go to A2.

A11) If $\Delta'=\Delta_J$, let $G^{k+1}=G^k-(K_J,L_J)$ and $k=k+1$. Go to A2.

**Theorem 3:** Algorithm A stops after finitely many steps. The output is a feasible basis different from the input, with associated objective function at least as good.

**Proof:** $\Delta'$ and (K,L) are well defined in A4 and A5 because row I is included in $T_I$ and therefore $\Delta'$ cannot exceed $b_I$. If I and J are included in the same subtree, then the reduced cost is negative when $\Delta=0$ and proportional to $Z(T,\Delta)$, therefore negative for all $\Delta$. If I and J are in different trees, then the reduced cost is negative for $\Delta\leq\Delta^*$, since the functions in A7 are continuous in $\Delta$. Since $\Delta'\leq\Delta^*$ in A8, the reduced cost is negative throughout and there is no possibility that the objective function might increase. It is also clear that G' is different from G, since G' includes (I,J) and G does not. The only question is whether A9 is encountered after finitely many steps.

When an edge is deleted, the effect on the basis is always that there is one more tree component and one less edge. Since A2 can only be revisited after deleting an edge, the number of edges in $G^0$ is an upper bound on the number of times A2 is encountered. Since A2 is encountered as frequently as any other step, the proof is complete. QED

The pivoting operation described above differs from the comparable Linear Programming operation in that the number of edges that have to leave the basis in order to get (I,J) in is not always 1; it may be any nonnegative integer. It differs essentially in this respect from Zangwill's (1970) Convex Simplex Method (CSM). In the CSM the number of basic variables is constant, and every pivot replaces a basic variable with a nonbasic variable. When a local minimum is encountered in CSM, the corresponding variable remains nonbasic, but positive, and consequently there is no unique solution associated with a given basis. In the method discussed above, the number of basic variables is not constant, but the idea that nonbasic variables are zero is preserved, along with the one-to-one relationship between bases and solutions. This relationship implies that the number of pivots is bounded by the number of bases, whereas the number of pivots required by the CSM may be unbounded.

## 5. Implementation in the Exponential Case

Two FORTRAN programs have been written that implement the methods of sections 3 and 4 in the Exponential case: MANIFO in the case of Manifold suboptimization and SIMPLX for the Extended Simplex Method. Each of these programs exploits the forest structure of the basis to minimize storage and facilitate computations. The forest is actually represented as a single tree by introducing an "earth" node to which all of the components of the basis are connected through a root node in each tree; however, in this exposition the term "tree" will continue to mean a component of the basis as heretofore. Most of the operations associated with pivoting are similar to those encountered in transshipment problems. Since these are well discussed in Bradley, Brown, and Graves (1977), only the aspects associated with the nonlinear nature of the objective function will be discussed here.

Consider first the problem of deleting an edge from the basis. If the edge is in tree T, then deleting it amounts to pruning off a branch from T. A reduced version of T will remain, along with a new tree T' consisting of the pruned off branch. Although equation (1) will hold in all trees before the pruning, it generally will not hold in either T or T' afterwards. The multipliers $(\lambda,\mu)$ must be scaled to make (1) hold in T and T', bearing in mind that y depends on $\mu$ and that nodes in T will have a different z-factor than nodes in T'. These z-factors can be computed using the formula given in Example 1 of Section 2. The computation in T' requires the sum $\sum_{j\varepsilon C(T')} \mu_j$. This number could be computed by summing $\mu_j$ over the appropriate set, but in fact both MANIFO and SIMPLX utilize an additional array SDM(k) to store the sum of $\mu_j$ over the set consisting of k and all of its column successors in whatever (rooted) tree k belongs to. Thus, if (I,J) is deleted from a tree with root node K, and if (say) J is the predecessor of I, then the required sum in T' is SDM(I) and the required sum in T is SDM(K)–SDM(I). Calculating the z-factor then requires a single exponentiation in each tree. After updating $(\lambda,\mu,y)$, the allocations in T and T' are computed as described in part b) of Theorem 2.; a preorder traversal array permits this to be done in one pass. The allocations themselves are stored in array X, with X(k) representing the allocation from node k to its predecessor if k is a row, or from the predecessor to k if k is a column. All arrays are of length n+m+1 in both programs (node length); in fact, the only function requiring mn operations is that of determining the entering edge by searching for the largest ratio $\mu_j E(i,j)/\lambda_i$ .

The addition of edge (I,J) to the basis requires the joining of one tree to another, possibly after or in conjunction with another pruning operation. If T' containing J is to be joined to T, T' is first "rehung" so that the root node is J. Next T' is scaled so that $\lambda_I=\mu_J E(I,J)$, which requires a logarithm in updating the potentials in T'. The unified tree is then scaled as in the previous paragraph. It is during the subsequent calculations of X that the possibility of negative allocations must be provided for in MANIFO.

SIMPLX uses all of the arrays in MANIFO plus one additional. The additional array W is associated with determining how big $\Delta$ can be before some basic variable becomes 0, as will be explained shortly. In the Exponential case the function $Z(T,\Delta)$ satisfies $\ln(Z(T,\Delta))=D(T)\Delta$, where

$$D(T)=[\lambda_I*\delta(T,I)-\mu_J*E(I,J)\delta(T,J)]/\sum_{j\varepsilon C(T)} \mu_j. \tag{7}$$

Note that D(T) is easy to compute because the denominator is already stored, and also that D(T) will be negative in step A5, positive in A6, and negative (since the reduced cost is negative) in A4. Multiplying $\mu_j$ by $Z(T,\Delta)$ is equivalent to subtracting $\ln(Z(T,\Delta))$ from $y_j$, so $y_j(\Delta)=y_j(0)-D(T)\Delta$ for $j\varepsilon C(T)$. Since this is a linear function of $\Delta$, allocation $x_{ij}(\Delta)$ will also be a linear function of $\Delta$ for all edges in T. The array element W(k) contains the rate at which the allocation associated with node k decreases with $\Delta$. The calculation of W is equivalent in difficulty to the calculation of X in MANIFO. Given W, it is simple to obtain the limiting value Δ' by comparing X/W ratios. The X array can be updated once Δ' is known.

Both procedures require a basis for a starting point. This is obtained by first including all edges of the form (k,k). If there are more columns than rows, this will leave some trees consisting of singleton columns. If there are more rows than columns, then row n+k is assigned to column k for $1 \le k \le n$, etc., until finally each row has been included in some edge. The resulting basis will be feasible, but unfortunately some edge (i,j) for which E(i,j) =0 may be included. The simplest (and the operative) remedy is to simply change E(i,j) to some small positive number, since in any case either procedure will quickly delete such an edge.

There is potential for roundoff errors to accumulate in the repeated updating of floating point arrays, so MANIFO and SIMPLX both use double precision arithmetic. Neither program incorporates any protection against degeneracy, but so far there have been no instances of cycling.

To give the reader a better idea of the actual operation of the algorithms, Figure 1 shows the sequence of bases considered by MANIFO in the process of solving a problem where m=3 and n=4. The basis is shown as a single tree with row nodes being distinguished from column nodes (as they are in MANIFO) by representing row nodes with negative numbers. Node 0 is "earth". With the convention of including all edges incident to node 0, a basis always has exactly n+m edges. Although X is stored as a node length array in MANIFO, the allocations are shown in figure 1 in the more familiar matrix form. There are $(2^4-1)^3=3375$ bases for this problem, of which MANIFO considers 8 feasible and 2 nonfeasible before finding one that is optimal. In the course of doing this MANIFO requires 18 logarithms or exponentials. These statistics are the same for SIMPLX in this small problem; differences emerge only in problems where MANIFO introduces more than 1 negative allocation at a time.

## 6. Computational Comparisons

MANIFO and SIMPLX were compared on randomly generated problems to determine which program is fastest. Problems generated were such that

E(i,j) is uniform [0,1]
$b_i$ is uniform [0,10]
V(j) is uniform [0,100]

Eleven (m,n) pairs were tested: (10,10), (7,20), (20,7), (20,20), (30,30), 40,40), (20,40), (40,20), (50,50), (60,60), and (70,70). Each of eleven (E,b) configurations was combined with each of five V configurations to generate a problem. For each of these 55 problems, the number of nonlinear operations (logarithms and exponentials) NL was recorded along with the CPU time T in seconds required to set up the initial guess and solve the problem. m, n, NL, and T were then transformed by taking logarithms and a linear regression performed.

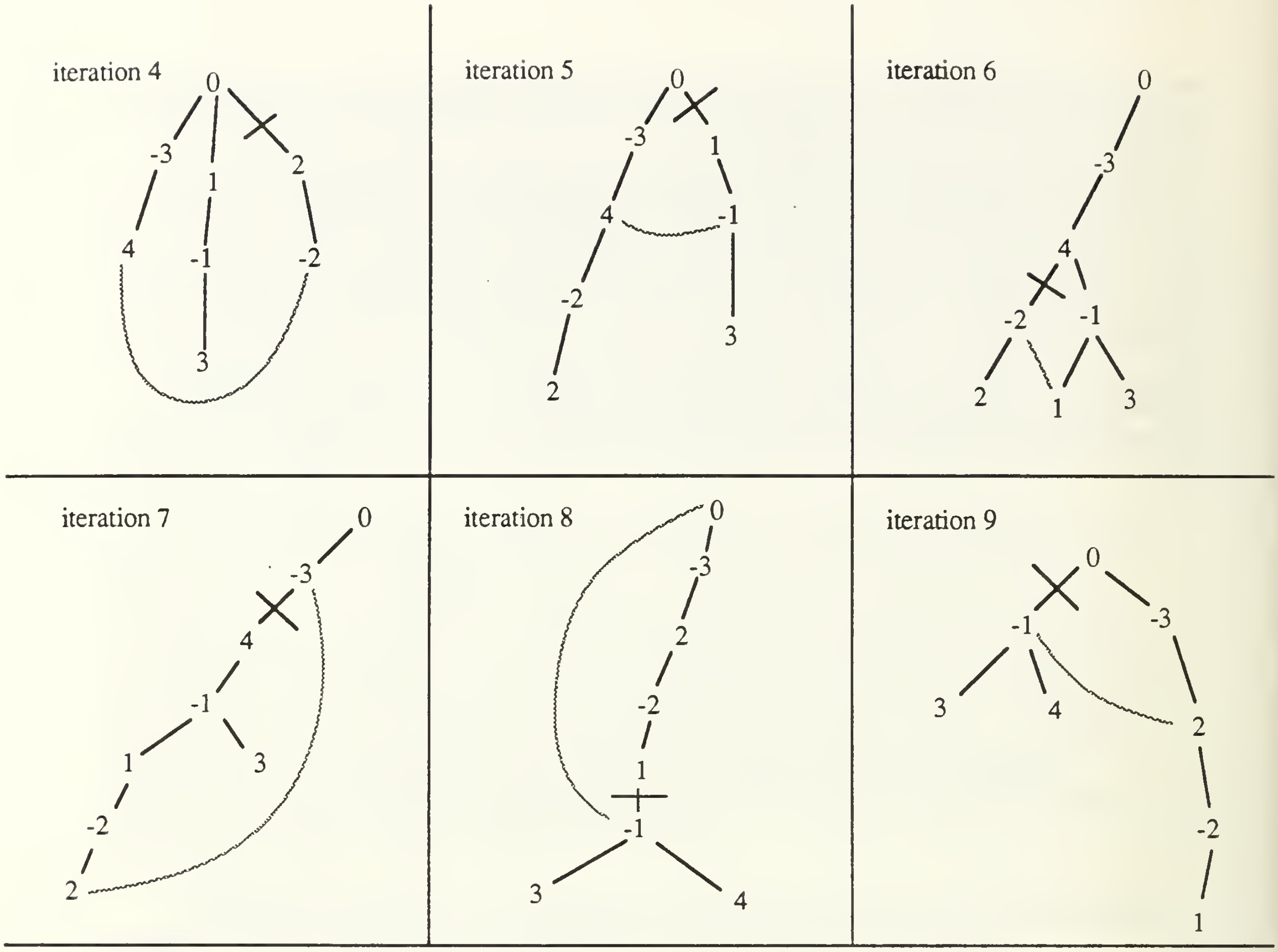
There turns out to be very little difference between MANIFO and SIMPLX. The fitted formulas are in each case approximately

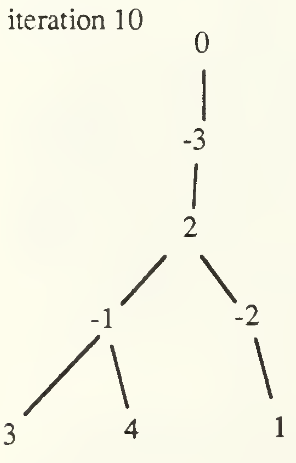
$$T = (.344\ \text{sec})(m/46)^{1.05}(n/122)^{.8} \exp(m/46+n/122) \quad (R^2=99\%) \qquad (8)$$

$$NL = 1.4\, m\sqrt{n} \quad (R^2 = 98\%) \qquad (9)$$

All runs were made on the Naval Postgraduate School's IBM3033AP. Since GAMS/MINOS is also available on that system, in one instance a problem with m=20 and n=20 was solved using GAMS/MINOS. The time required was 1.88 seconds ("MINOS TIME" in the output), which is 47 times as large as the time required by MANIFO (.04 seconds) for the same problem. Further comparisons were not made because MINOS is a general purpose solver that does not exploit the network nature of the constraints. A better comparison would be with the GENOS network optimizer (Mulvey and Zenios (1987)). Attempts to accomplish this are in progress, but so far the lack of an exponential function in GENOS 1.0 has proved to be a roadblock.

The reason for keeping track of the number of exponentials and logarithms NL in running MANIFO/SIMPLX is that it was initially anticipated that such nonlinear operations would require relatively large amounts of time. This turns out not to be the case. When (m,n) = (70,70), formulas (8) and (9) predict T = .28 seconds and NL = 820. Since an IBM3033AP requires only about 4μsec for a double precision logarithm or exponential, the amount of time spent in nonlinear operations is only about $(820)(4\times10^{-6}) = .0033$ seconds; MANIFO and SIMPLX are each mainly occupied with manipulating and scaling the various arrays required to represent the current basic solution as a tree, rather than solving the balance equation. This is encouraging for applications where the objective function, although "good," might require a numeric rather than analytic solution of the balance equation.





μ

| λ | .0250 | .0375 | .0250 | .0187 |
|---|---|---|---|---|
| .0750 | | .372 | 1.461 | 1.167 |
| .0750 | 1.230 | .770 | | |
| .0375 | | 1 | | |
| y | 3.689 | 3.284 | 4.382 | 4.670 |

$\boxed{\text{optimal solution: } f(y) = .1062}$

**Figure 1**
**Solution of a 3x4 Problem Using MANIFO (Continued)**

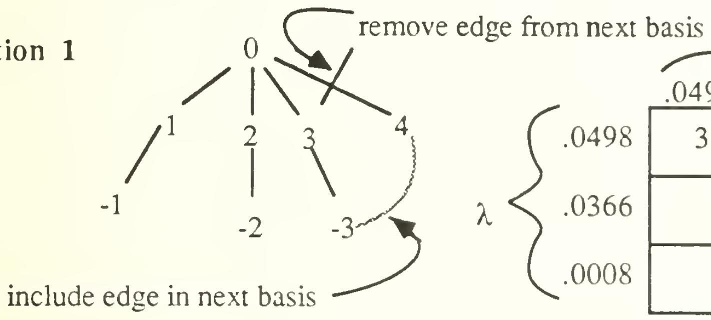
**PROBLEM DATA**

input E(i,j) shown in matrix

$$\overbrace{\begin{array}{cccc|c} 1 & 1 & 2 & 2 & \\ \hline 1 & 2 & 3 & 4 & 3 \\ 3 & 2 & 2 & 1 & 2 \\ .0004 & 1 & .0004 & 1 & 1 \end{array}}^{\text{input V}} \quad \left.\begin{array}{c} 3 \\ 2 \\ 1 \end{array}\right\} \text{input b}$$

---

**iteration 1**

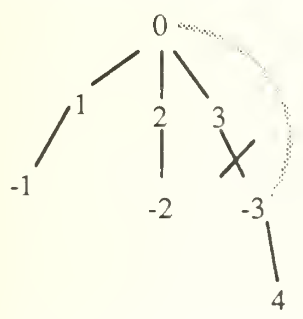


$$\begin{array}{c|cccc} & \multicolumn{4}{c}{\mu} \\ & .0498 & .0183 & 1.999 & 2.000 \\ \hline \lambda\ .0498 & 3 & & & \\ .0366 & & 2 & & \\ .0008 & & & 1 & \\ \hline & .0498 & .0183 & 1.999 & 2.000 \\ & \multicolumn{4}{c}{y} \end{array}$$

$\boxed{\text{starting solution: } f(y) = 4.067}$

---

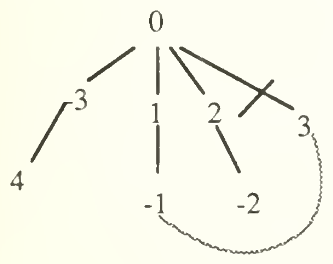
**iteration 2**



$$\begin{array}{c|cccc} & .0498 & .0183 & 2.005 & .0008 \\ \hline .0498 & 3 & & & \\ .0366 & & 2 & & \\ .0008 & & & -6.821 & 7.821 \\ \hline & 3 & 4 & -.0027 & 7.821 \end{array}$$

---

**iteration 3**



$$\begin{array}{c|cccc} & .0498 & .0183 & 2.000 & .7358 \\ \hline .0498 & 3 & & & \\ .0366 & & 2 & & \\ .7358 & & & & 1 \\ \hline & 3 & 4 & 0 & 1 \end{array}$$

---

Figure 1.
Solution of a 3x4 Problem Using MANIFO

## References

1. Ahlfeld, D., R. Dembo, J. Mulvey, and S. Zenios. 1987. Nonlinear programming on generalized networks. *ACM Transactions on Mathematical Software* **13,** 350-367.
2. Bazaraa, M., and C. Shetty. 1979. Nonlinear programming:theory and algorithms. Wiley, page 503.
3. Berge, C. 1962. *The Theory of Graphs.* Wiley, page 152.
4. Bradley, G., G. Brown, and G. Graves. 1977. Design and implementation of large scale primal transshipment problems. *Management Science* **24**, 1-34.
5. Clasen, R., G. Graves, and J. Lu. 1974. Sortie allocation by a nonlinear programming model for determining a munitions mix. RAND R-1411-DDPAE.
6. Koopman, B. 1956. Theory of search II, *Opns. Res.* **4**, 519-521.
7. Washburn, A. 1981. Note on constrained maximization of a sum. *Opns. Res.* **29**, 411-414.
8. Zangwill, W. 1970. *Nonlinear programming: a unified approach.* Prentice-Hall, chapter 8.

## Initial Distribution List

| | |
|---|---|
| Library (Code 0142)<br>Naval Postgraduate School<br>Monterey, CA 93943-5000 | 2 |
| Defense Technical Information Center<br>Cameron Station<br>Alexandria, VA 22314 | 2 |
| Stavros Zenios<br>Decision Sciences Dept.<br>The Wharton School<br>U. of Pennsylvania<br>Philadelphia, PA 19104 | 1 |
| Willard Zangwill<br>Graduate School of Business<br>U. of Chicago<br>Chicago, IL 60637 | 1 |
| Richard Helgason<br>IE/OR Dept.<br>School of Engineering and Applied Science<br>Southern Methodist University<br>Dallas, TX 75275 | 1 |
| Jeff Kennington<br>IE/OR Dept.<br>School of Engineering and Applied Science<br>Southern Methodist University<br>Dallas, TX 75275 | 1 |
| Ron Dembo<br>398 Markham St.<br>Toronto, Ontario M6G 2K9<br>CANADA | 1 |
| Leon Lasdon<br>MS/IS Dept.<br>School of Business Administration<br>U. of Texas<br>Austin, TX 78712 | 1 |
| Mike Engquist<br>Cleveland Consulting Assoc.<br>3415 Graystone Suite 204<br>Austin, TX 78731 | 1 |

Richard McBride 1
Decision Systems Department
UCLA
Los Angeles, CA 90089-1421

Don Hearn 1
Dept. of Ind. Sys. Engr.
303 Weil Hall
U. of Florida
Gainesville, FL 32611

John Klincewicz 1
3L-101
AT&T Bell Laboratories
Crawfords Corner Road
Holmdel, NJ 07733

David Ahlfeld 1
P. O. Box 1521
Princeton, NJ 08542

John M. Mulvey 1
Engineering-Management Systems Prog.
Civil Engineering Department
Princeton University
Princeton, NJ 08544

Prof. G. Bradley, Code 55Bz 1
Naval Postgraduate School
Monterey, CA 93943-5000

Prof. R. Wood, Code 55Wd 1
Naval Postgraduate School
Monterey, CA 93943-5000

Prof. G. Brown, Code 55Bw 1
Naval Postgraduate School
Monterey, CA 93943-5000

Prof. S. Lawphongpanich, Code 55Lp 1
Naval Postgraduate School
Monterey, CA 93943-5000

Prof. R. Rosenthal, Code 55Rl 1
Naval Postgraduate School
Monterey, CA 93943-5000

Prof. J. Eagle, Code 55Er 1
Naval Postgraduate School
Monterey, CA 93943-5000

| | |
|---|---|
| Prof. A. Washburn, Code 55Ws<br>Naval Postgraduate School<br>Monterey, CA 93943-5000 | 10 |
| Research Administration<br>Code: 012<br>Naval Postgraduate School<br>Monterey, CA 93943 | 1 |